WEST

Help | Logout

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Your wildcard search against 2000 terms has yielded the results below

Search for additional matches among the next 2000 terms

Generate Collection

Search Results - Record(s) 1 through 10 of 23 returned.

□ 1. Document ID: US 5892900 A Relevance Rank: 72

```
Entry 7 of 23                     File: USPT                     Apr 6, 1999

US-PAT-NO: 5892900
DOCUMENT-IDENTIFIER: US 5892900 A
TITLE: Systems and methods for secure transaction management and electronic rights
protection
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 2. Document ID: US 5915019 A Relevance Rank: 72

```
Entry 5 of 23                     File: USPT                     Jun 22, 1999

US-PAT-NO: 5915019
DOCUMENT-IDENTIFIER: US 5915019 A
TITLE: Systems and methods for secure transaction management and electronic rights
protection
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 3. Document ID: US 5949876 A Relevance Rank: 71

```
Entry 2 of 23                     File: USPT                     Sep 7, 1999

US-PAT-NO: 5949876
DOCUMENT-IDENTIFIER: US 5949876 A
TITLE: Systems and methods for secure transaction management and electronic rights
protection
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 4. Document ID: US 5982891 A Relevance Rank: 71

```
Entry 1 of 23                     File: USPT                     Nov 9, 1999
```

stantly maintained and available for use by any OPEN method that starts. In other implementations, the channel for open processing may be rebuilt and restarted each time an OPEN method starts.

Read

FIG. **50**, **50***a*–**50***f* show examples of process control steps for performing a representative example of a READ method **1650**. Comparing FIG. **50** with FIG. **49** reveals that the same overall high level processing may typically be performed for READ method **1650** as was described in connection with OPEN method **1500**. Thus, READ method **1650** may call a control method **1652** in response to a read event, the control method in turn invoking an EVENT method **1654**, a METER method **1656**, a BILLING method **1658** and a BUDGET method **1660**. In the preferred embodiment, READ control method **1652** may request methods to fingerprint and/or obscure content before releasing the decrypted content.

FIGS. **50***a*–**50***e* are similar to FIGS. **49***a*–**49***e*. Of course, even though the same user data elements may be used for both the OPEN method **1500** and the READ method **1650**, the method data elements for the READ method may be completely different, and in addition, the user data elements may provide different auditing, metering, billing and/or budgeting criteria for read as opposed to open processing.

Referring to FIG. **50***f*, the READ control method **1652** must determine which key to use to decrypt content if it is going to release decrypted content to the user (block **1758**). READ control method **1652** may make this key determination based, in part, upon the PERC **808** for the object (block **1760**). READ control method **1652** may then call an ACCESS method to actually obtain the encrypted content to be decrypted (block **1762**). The content is then decrypted using the key determined by block **1758** (block **1764**). READ control method **1652** may then determine whether a "fingerprint" is desired (decision block **1766**). If fingerprinting of the content is desired ("yes" exit of decision block **1766**), READ control method **1652** may call the FINGERPRINT method (block **1768**). Otherwise, READ control method **1652** may determine whether it is desired to obscure the decrypted content (decision block **1770**). If so, READ control method **1652** may call an OBSCURE method to perform this function (block **1772**). Finally, READ control method **1652** may commit the secure database transaction (block **1774**), optionally tear down the read channel (not shown), and terminate (block **1776**).

Write

FIGS. **51**, **51***a*–**51***f* are flowcharts of examples of process control steps used to perform a representative example of a WRITE method **1780** in the preferred embodiment. WRITE method **1780** uses a control method **1782** to call an EVENT method **1784**, METER method **1786**, BILLING method **1788**, and BUDGET method **1790** in this example. Thus, writing information into a container (either by overwriting information already stored in the container or adding new information to the container) in the preferred embodiment may be metered, billed and/or budgeted in a manner similar to the way opening a container and reading from a container can be metered, billed and budgeted. As shown in FIG. **51**, the end result of WRITE method **1780** is typically to encrypt content, update the container table of contents and related information to reflect the new content, and write the content to the object.

FIG. **51***a* for the WRITE control method **1782** is similar to FIG. **49***a* and FIG. **50***a* for the OPEN control method and the READ control method, respectively. However, FIG. **51***b* is slightly different from its open and read counterparts. In particular, block **1820** is performed if the WRITE EVENT method **1784** fails. This block **1820** updates the EVENT method map MDE to reflect new data. This is necessary to allow information written by block **1810** to be read by FIG. **51***b* READ method block **1678** based on the same (but now updated) EVENT method map MDE.

Looking at FIG. **51***f*, once the EVENT, METER, BILLING and BUDGET methods have returned successfully to WRITE control method **1782**, the WRITE control method writes audit information to Audit UDE (blocks **1890**, **1892**), and then determines (based on the PERC for the object and user and an optional algorithm) which key should be used to encrypt the content before it is written to the container (blocks **1894**, **1896**). CONTROL method **1782** then encrypts the content (block **1898**) possibly by calling an ENCRYPT method, and writes the encrypted content to the object (block **1900**). CONTROL method **1782** may then update the table of contents (and related information) for the container to reflect the newly written information (block **1902**), commit the secure database transaction (block **1904**), and return (block **1906**).

Close

FIG. **52** is a flowchart of an example of process control steps to perform a representative example of a CLOSE method **1920** in the preferred embodiment. CLOSE method **1920** is used to close an open object. In the preferred embodiment, CLOSE method **1920** primes an audit trail and writes audit information to an Audit UDE (blocks **1922**, **1924**). CLOSE method **1920** then may destroy the current channel(s) being used to support and/or process one or more open objects (block **1926**). As discussed above, in some (e.g., multi-user or multi-tasking) installations, the step of destroying a channel is not needed because the channel may be left operating for processing additional objects for the same or different users. CLOSE method **1920** also releases appropriate records and resources associated with the object at this time (block **1926**). The CLOSE method **1920** may then write an audit trail (if required) into an Audit UDE (blocks **1928**, **1930**) before completing.

Event

FIG. **53***a* is a flowchart of example process control steps provided by a more general example of an EVENT method **1940** provided by the preferred embodiment. Examples of EVENT methods are set forth in FIGS. **49***b*, **50***b* and **51***b* and are described above. EVENT method **1940** shown in FIG. **53***a* is somewhat more generalized than the examples above. Like the EVENT method examples above, EVENT method **1940** receives an identification of the event along with an event count and event parameters. EVENT method **1940** may first prime an EVENT audit trail (if required) by writing appropriate information to an EVENT method Audit Trail UDE (blocks **1942**, **1944**). EVENT method **1940** may then obtain and load an EVENT method map DTD from the secure database (blocks **1946**, **1948**). This EVENT method map DTD describes, in this example, the format of the EVENT method map MDE to be read and accessed immediately subsequently (by blocks **1950**, **1952**). In the preferred embodiment, MDEs and UDEs may have any of various different formats, and their formats may be flexibly specified or changed dynamically depending upon the installation, user, etc. The DTD, in effect, describes to the EVENT method **1940** how to read from the EVENT method

US-PAT-NO: 5982891
DOCUMENT-IDENTIFIER: US 5982891 A
TITLE: Systems and methods for secure transaction management and electronic rights protection

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 5. Document ID: US 5917912 A Relevance Rank: 71

Entry 4 of 23 File: USPT Jun 29, 1999

US-PAT-NO: 5917912
DOCUMENT-IDENTIFIER: US 5917912 A
TITLE: System and methods for secure transaction management and electronic rights protection

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 6. Document ID: US 5910987 A Relevance Rank: 71

Entry 6 of 23 File: USPT Jun 8, 1999

US-PAT-NO: 5910987
DOCUMENT-IDENTIFIER: US 5910987 A
TITLE: Systems and methods for secure transaction management and electronic rights protection

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 7. Document ID: US 5630057 A Relevance Rank: 68

Entry 13 of 23 File: USPT May 13, 1997

US-PAT-NO: 5630057
DOCUMENT-IDENTIFIER: US 5630057 A
TITLE: Secure architecture and apparatus using an independent computer cartridge

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 8. Document ID: US 5442541 A Relevance Rank: 68

Entry 17 of 23 File: USPT Aug 15, 1995

US-PAT-NO: 5442541
DOCUMENT-IDENTIFIER: US 5442541 A
TITLE: Enabling features over common communication channel

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 9. Document ID: US 5499295 A Relevance Rank: 68

Entry 15 of 23 File: USPT Mar 12, 1996

ing" the changes made to audit trail UDE by blocks **1540**, **1548**. However, this "roll back" performed by block **1554** in the preferred embodiment does not "undo" the changes made to the control method audit UDE by blocks **1532**, **1534**.

Assuming the EVENT metbod **1504** completed successfully, control method **1502** then calls the METER method **1506** shown on FIG. **49***c*. In the preferred embodiment, METER method **1506** primes the meter audit trail if required (block **1558**), which typically involves writing to a METER method audit trail UDE (block **1560**). METER method **1506** may then read a METER method UDE from the secure database (block **1562**), modify the meter UDE by adding an appropriate event count to the meter value contained in the meter UDE (block **1564**), and then writing the modified meter UDE back to the secure database (block **1562**). In other words, block **1564** may read the meter UDE, increment the meter count it contains, and write the changed meter UDE back to the secure database. In the preferred embodiment, METER method **1506** may then write meter audit trail information to the METER method audit trail UDE if required (blocks **1566**, **1568**). METER method **1506** preferably next performs a test to determine whether the meter increment succeeded (decision block **1570**). METER metbod **1506** returns to control method **1502** with a completion code (e.g., succeed or fail) and a meter value determined by block **1564**.

Control method **1502** tests whether the METER metbod succeeded by examining the completion code, for example (decision block **1572**). If the METER method failed ("no" exit to decision block **1572**), then control method **1502** "rolls back" a secure database transaction (block **1574**), and returns with an indication that the OPEN method failed (block **1576**). Assuming the METER metbod succeeded ("yes" exit to decision block **1572**), control method **1502** calls the BILLING method **1508** and passes it the meter value provided by METER method **1506**.

An example of steps performed by BILLING method **1508** is set forth in FIG. **49***d*. BILLING method **1508** may prime a billing audit trail if required (block **1578**) by writing to a BILLING method Audit Trail UDE within tbe secure database (block **1580**). BILLING method **1508** may then map the atomic element number, count and meter value to a billing amount using a BILLING method map MDE read from the secure database (blocks **1582**, **1584**). Providing an independent BILLING method map MDE containing, for example, price list information, allows separately deliverable pricing for the billing process. The resulting billing amount generated by block **1582** may be written to the BILLING method Audit Trail UDE (blocks **1586**, **1588**), and may also be returned to control method **1502**. In addition, BILLING method **1508** may determine wbether a billing amount was properly selected by block **1582** (decision block **1590**). In this example, the test performed by block **1590** generally requires more than mere examination of the returned billing amount, since the billing amount may be changed in unpredictable ways as specified by BILLING method map MDE. Control then returns to control method **1502**, which tests the completion code provided by BILLING method **1508** to determine whether the BILLING method succeeded or failed (block **1592**). If the BILLING method failed ("no" exit to decision block **1592**), control method **1502** may "roll back" the secure database transaction (block **1594**), and return an indication that the OPEN method failed (block **1596**). Assuming the test performed by decision block **1592** indicates that the BILLING method succeeded ("yes" exit to decision block **1592**), then control method **1502** may call BUDGET method **1510**.

Other BILLING methods may use site, user and/or usage information to establish, for example, pricing information. For example, information concerning the presence or absence of an object may be used in establisbing "suite" purchases, competitive discounts, etc. Usage levels may be factored into a BILLING method to establish price breaks for different levels of usage. A currency translation feature of a BILLING method may allow purchases and/or pricing in many different currencies. Many other possibilities exist for determining an amount of budget consumed by an event that may be incorporated into BILLING methods.

An example of detailed control steps performed by BUDGET method **1510** is set forth in FIG. **49***e*. BUDGET method **1510** may prime a budget audit trail if required by writing to a budget trail UDE (blocks **1598**, **1600**). BUDGET method **1510** may next perform a billing operation by adding a billing amount to a budget value (block **1602**). This operation may be performed, for example, by reading a BUDGET method UDE from the secure database, modifying it, and writing it back to the secure database (block **1604**). BUDGET method **1510** may then write the budget audit trail information to the BUDGET method Audit Trail UDE (blocks **1606**, **1608**). BUDGET method **1510** may finally, in this example, determine whether the user has run out of budget by determining whetber the budget value calculated by block **1602** is out of range (decision block **1610**). If the user has run out of budget ("yes" exit to decision block **1610**), the BUDGET method **1510** may return a "fail completion" code to control method **1502**. BUDGET method **1510** then returns to control method **1502**, wbich tests whether the BUDGET method completion codc was successful (decision block **1612**). If the BUDGET method failed ("no" exit to decision block **1612**), control method **1502** may "roll back" thc secure database transaction and itself return with an indication that the OPEN metbod failed (blocks **1614**, **1616**). Assuming control metbod **1502** determines that the BUDGET method was successful, the control method may perform the additional steps shown on FIG. **49***f*. For example, control method **1502** may write an open audit trail if required by writing audit information to the audit UDE that was primed at block **1532** (blocks **1618**, **1620**). Control method **1502** may then establish a read event processing (block **1622**), using the User Right Table and the PERC associated with the object and user to establish the channel (block **1624**). This channel may optionally be shared between users of the VDE node **600**, or may be used only by a specified user.

Control method **1502** tben, in the preferred embodiment, tests whether the read channel was established successfully (decision block **1626**). If the read channel was not successfully establisbed ("no" exit to decision block **1626**), control method **1502** "rolls back" the secured database transaction and provides an indication that the OPEN method failed (blocks **1628**, **1630**). Assuming the read channel was successfully established ("yes" exit to decision block **1626**), control method **1502** may "commit" the secure database transaction (block **1632**). This step of "committing" the secure database transaction in the preferred embodiment involves, for example, deleting intermediate values associated with the secure transaction that has just been performed and, in one example, writing changed UDEs and MDEs to the secure database. It is generally not possible to "roll back" a secure transaction once it has been committed by block **1632**. Then, control method **1502** may "tear down" the channel for open processing (block **1634**) before terminating (block **1636**). In some arrangements, such as multi-tasking VDE node environments, the open channel may be con-

```
US-PAT-NO: 5499295
DOCUMENT-IDENTIFIER: US 5499295 A
TITLE: Method and apparatus for feature authorization and software copy protection
in RF communications devices
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 10. Document ID: US 4864494 A Relevance Rank: 68

```
Entry 23 of 23                          File: USPT                     Sep 5, 1989

US-PAT-NO: 4864494
DOCUMENT-IDENTIFIER: US 4864494 A
TITLE: Software usage authorization system with key for
decrypting/re-encrypting/re-transmitting moving target security codes from
protected software
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

Generate Collection

Terms	Documents
(copy protect$) and cryptograph$ and (control$ with copy$ with function$)	23

Display 10 Documents including document number 11

Display Format: TI Change Format

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Help Logout

and whether the open event is significant in the sense that it needs to be processed by METER method **1506**, BILLING method **1508**, and/or BUDGET method **1510**. EVENT method **1504** may maintain audit trail information within an audit trail UDE, and may determine permissions and significance of the event by using an Event Method Data Element (MDE). EVENT method **1504** may also map the open event into an "atomic element" and count that may be processed by METER method **1506**, BILLING method **1508**, and/or BUDGET method **1510**.

In OPEN method **1500**, once EVENT method **1504** has been called and returns successfully, control method **1502** then may call METER method **1506** and pass the METER method, the atomic element and count returned by EVENT method **1504**. METER method **1506** may maintain audit trail information in a METER method Audit Trail UDE, and may also maintain meter information in a METER method UDE. In the preferred embodiment, METER method **1506** returns a meter value to control method **1502** assuming successful completion.

In the preferred embodiment, control method **1502** upon receiving an indication that METER method **1506** has completed successfully, then calls BILLING method **1508**. Control method **1502** may pass to BILLING method **1508** the meter value provided by METER method **1506**. BILLING method **1508** may read and update billing information maintained in a BILLING method map MDE, and may also maintain and update audit trail in a BILLING method Audit Trail UDE. BILLING method **1508** may return a billing amount and a completion code to control method **1502**

Assuming BILLING method **1508** completes successfully, control method **1502** may pass the billing value provided by BILLING method **1508** to BUDGET method **1510**. BUDGET method **1510** may read and update budget information within a BUDGET method UDE, and may also maintain audit trail information in a BUDGET method Audit Trail UDE. BUDGET method **1510** may return a budget value to control method **1502**, and may also return a completion code indicating whether the open event exceeds the user's budget (for this type of event).

Upon completion of BUDGET method **1510**, control method **1502** may create a channel and establish read/use control information in preparation for subsequent calls to the READ method.

FIGS. **49***a*–**49***f* are a more detailed description of the OPEN method **1500** example shown in FIG. **49**. Referring to FIG. **49***a*, in response to an open event, control method **1502** first may determine the identification of the object to be opened and the identification of the user that has requested the object to be opened (block **1520**). Control method **1502** then determines whether the object to be opened is registered for this user (decision block **1522**). It makes this determination at least in part in the preferred embodiment by reading the PERC **808** and the User Rights Table (URT) element associated with the particular object and particular user determined by block **1520** (block **1524**). If the user is not registered for this particular object ("no" exit to decision block **1522**), then control method **1502** may call the REGISTER method for the object and restart the OPEN method **1500** once registration is complete (block **1526**). The REGISTER method block **1526** may be an independent process and may be time independent. It may, for example, take a relatively long time to complete the REGISTER method (say if the VDE distributor or other participant responsible for providing registration wants to perform a credit check on the user before registering the user for this particular object).

Assuming the proper URT for this user and object is present such that the object is registered for this user ("yes" exit to decision block **1522**), control method **1502** may determine whether the object is already open for this user (decision block **1528**). This test may avoid creating a redundant channel for opening an object that is already open. Assuming the object is not already open ("no" exit to decision block **1528**), control method **1502** creates a channel and binds appropriate open control elements to it (block **1530**). It reads the appropriate open control elements from the secure database (or the container, such as, for example, in the case of a travelling object), and "binds" or "links" these particular appropriate control elements together in order to control opening of the object for this user. Thus, block **1530** associates an event with one or more appropriate method core(s), appropriate load modules, appropriate User Data Elements, and appropriate Method Data Elements read from the secure database (or the container) (block **1532**). At this point, control method **1502** specifies the open event (which started the OPEN method to begin with), the object ID and user ID (determined by block **1520**), and the channel ID of the channel created by block **1530** to subsequent EVENT method **1504**, METER method **1506**, BILLING method **1508** and BUDGET method **1510** to provide a secure database "transaction" (block **1536**). Before doing so, control method **1502** may prime an audit process (block **1533**) and write audit information into an audit UDE (block **1534**) so a record of the transaction exists even if the transaction fails or is interfered with.

The detail steps performed by EVENT method **1504** are set forth on FIG. **49***b*. EVENT method **1504** may first prime an event audit trail if required (block **1538**) which may write to an EVENT Method Audit Trail UDE (block **1540**). EVENT method **1504** may then perform the step of mapping the open event to an atomic element number and event count using a map MDE (block **1542**). The EVENT method map MDE may be read from the secure database (block **1544**). This mapping process performed by block **1542** may, for example, determine whether or not the open event is meterable, billable, or budgetable, and may transform the open event into some discrete atomic element for metering, billing and/or budgeting. As one example, block **1542** might perform a one-to-one mapping between open events and "open" atomic elements, or it may only provide an open atomic element for every fifth time that the object is opened. The map block **1542** preferably returns the open event, the event count, the atomic element number, the object ID, and the user ID. This information may be written to the EVENT method Audit Trail UDE (block **1546**, **1548**). In the preferred embodiment, a test (decision block **1550**) is then performed to determine whether the EVENT method failed. Specifically, decision block **1550** may determine whether an atomic element number was generated. If no atomic element number was generated (e.g., meaning that the open event is not significant for processing by METER method **1506**, BILLING method **1508** and/or BUDGET method **1510**), then EVENT method **1504** may return a "fail" completion code to control method **1502** ("no" exit to decision block **1550**).

Control method **1502** tests the completion code returned by EVENT method **1504** to determine whether it failed or was successful (decision block **1552**). If the EVENT method failed ("no" exit to decision block **1552**), control method **1502** may "roll back" the secure database transaction (block **1554**) and return itself with an indication that the OPEN method failed (block **1556**). In this context, "rolling back" the secure database transaction means, for example, "undo-

WEST

Help | Logout

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Your wildcard search against 2000 terms has yielded the results below

Search for additional matches among the next 2000 terms

Generate Collection

Search Results - Record(s) 11 through 20 of 23 returned.

□ 11. Document ID: US 5677953 A Relevance Rank: 68

```
Entry 11 of 23                     File: USPT                    Oct 14, 1997

US-PAT-NO: 5677953
DOCUMENT-IDENTIFIER: US 5677953 A
TITLE: System and method for access control for portable data storage media
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 12. Document ID: US 5457746 A Relevance Rank: 68

```
Entry 16 of 23                     File: USPT                    Oct 10, 1995

US-PAT-NO: 5457746
DOCUMENT-IDENTIFIER: US 5457746 A
TITLE: System and method for access control for portable data storage media
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 13. Document ID: US 5703951 A Relevance Rank: 68

```
Entry 10 of 23                     File: USPT                    Dec 30, 1997

US-PAT-NO: 5703951
DOCUMENT-IDENTIFIER: US 5703951 A
TITLE: System and method for access data control
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 14. Document ID: US 5220604 A Relevance Rank: 68

```
Entry 21 of 23                     File: USPT                    Jun 15, 1993

US-PAT-NO: 5220604
DOCUMENT-IDENTIFIER: US 5220604 A
TITLE: Method for performing group exclusion in hierarchical group structures
```

of SPU software/firmware. Such permanent portions may include, for example, code that interfaces to hardware elements such as the RTC 528, encryption/decryption engine 522, interrupt handlers, key generators, etc. Some of the operating system, library calls, libraries, and many of the core services provided by SPU 500 may also be in masked ROM 532*a*. In addition, some of the more commonly used executables are also good candidates for inclusion in masked ROM 532*a*. Items that need to be updated or that need to disappear when power is removed from SPU 500 should not be stored in masked ROM 532*a*.

Under some circumstances, RAM 534*a* and/or NVRAM 534*b* (NVRAM 534*b* may, for example, be constantly powered conventional RAM) may perform at least part of the role of ROM 532.

SPU Internal RAM

SPU 500 general purpose RAM 534 provides, among other things, secure execution space for secure processes. In the preferred embodiment, RAM 534 is comprised of different types of RAM such as a combination of high-speed RAM 534*a* and an NVRAM ("non-volatile RAM") 534*b*. RAM 534*a* may be volatile, while NVRAM 534*b* is preferably battery backed or otherwise arranged so as to be non-volatile (i.e., it does not lose its contents when power is turned off).

High-speed RAM 534*a* stores active code to be executed and associated data structures.

NVRAM 534*b* preferably contains certain keys and summary values that are preloaded as part of an initialization process in which SPU 500 communicates with a VDE administrator, and may also store changeable or changing information associated with the operation of SPU 500. For security reasons, certain highly sensitive information (e.g., certain load modules and certain encryption key related information such as internally generated private keys) needs to be loaded into or generated internally by SPU 500 from time to time but, once loaded or generated internally, should never leave the SPU. In this preferred embodiment, the SPU 500 non-volatile random access memory (NVRAM) 534*b* may be used for securely storing such highly sensitive information. NVRAM 534*b* is also used by SPU 500 to store data that may change frequently but which preferably should not be lost in a power down or power fail mode.

NVRAM 534*b* is preferably a flash memory array, but may in addition or alternatively be electrically erasable programmable read only memory (EEPROM), static RAM (SRAM), bubble memory, three dimensional holographic or other electro-optical memory, or the like, or any other writable (e.g., randomly accessible) non-volatile memory of sufficient speed and cost-effectiveness.

SPU External Memory

The SPU 500 can store certain information on memory devices external to the SPU. If available, electronic appliance 600 memory can also be used to support any device external portions of SPU 500 software. Certain advantages may be gained by allowing the SPU 500 to use external memory. As one example, memory internal to SPU 500 may be reduced in size by using non-volatile read/write memory in the host electronic appliance 600 such as a non-volatile portion of RAM 656 and/or ROM 658.

Such external memory may be used to store SPU programs, data and/or other information. For example, a VDE control program may be, at least in part, loaded into the memory and communicated to and decrypted within SPU 500 prior to execution. Such control programs may be re-encrypted and communicated back to external memory where they may be stored for later execution by SPU 500. "Kernel" programs and/or some or all of the non-kernel "load modules" may be stored by SPU 500 in memory external to it. Since a secure database 610 may be relatively large, SPU 500 can store some or all of secure database 610 in external memory and call portions into the SPU 500 as needed.

As mentioned above, memory external to SPU 500 may not be secure. Therefore, when security is required, SPU 500 must encrypt secure information before writing it to external memory, and decrypt secure information read from external memory before using it. Inasmuch as the encryption layer relies on secure processes and information (e.g., encryption algorithms and keys) present within SPU 500, the encryption layer effectively "extends" the SPU security barrier 502 to protect information the SPU 500 stores in memory external to it.

SPU 500 can use a wide variety of different types of external memory. For example, external memory may comprise electronic appliance secondary storage 652 such as a disk; external EEPROM or flash memory 658; and/or external RAM 656. External RAM 656 may comprise an external nonvolatile (e.g. constantly powered) RAM and/or cache RAM.

Using external RAM local to SPU 500 can significantly improve access times to information stored externally to an SPU. For example, external RAM may be used:

to buffer memory image pages and data structures prior to their storage in flash memory or on an external hard disk (assuming transfer to flash or hard disk can occur in significant power or system failure cases);

provide encryption and decryption buffers for data being released from VDE objects 300.

to cache "swap blocks" and VDE data structures currently in use as an aspect of providing a secure virtual memory environment for SPU 500.

to cache other information in order to, for example, reduce frequency of access by an SPU to secondary storage 652 and/or for other reasons.

Dual ported external RAM can be particularly effective in improving SPU 500 performance, since it can decrease the data movement overhead of the SPU bus interface unit 530 and SPU microprocessor 520.

Using external flash memory local to SPU 500 can be used to significantly improve access times to virtually all data structures. Since most available flash storage devices have limited write lifetimes, flash storage needs to take into account the number of writes that will occur during the lifetime of the flash memory. Hence, flash storage of frequently written temporary items is not recommended. If external RAM is non-volatile, then transfer to flash (or hard disk) may not be necessary.

External memory used by SPU 500 may include two categories:

external memory dedicated to SPU 500, and

memory shared with electronic appliance 600.

For some VDE implementations, sharing memory (e.g., electronic appliance RAM 656, ROM 658 and/or secondary storage 652) with CPU 654 or other elements of an electronic appliance 600 may be the most cost effective way to store VDE secure database management files 610 and information that needs to be stored external to SPU 500. A host system hard disk secondary memory 652 used for general purpose file storage can, for example, also be used to store VDE management files 610. SPU 500 may be given exclusive access to the external memory (e.g., over a local bus high speed connection provided by BIU 530). Both dedicated and shared external memory may be provided.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 15. Document ID: US 5224163 A Relevance Rank: 68

Entry 20 of 23 File: USPT Jun 29, 1993

US-PAT-NO: 5224163
DOCUMENT-IDENTIFIER: US 5224163 A
TITLE: Method for delegating authorization from one entity to another through the use of session encryption keys

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 16. Document ID: US 5315657 A Relevance Rank: 68

Entry 19 of 23 File: USPT May 24, 1994

US-PAT-NO: 5315657
DOCUMENT-IDENTIFIER: US 5315657 A
TITLE: Compound principals in access control lists

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 17. Document ID: US 5734721 A Relevance Rank: 68

Entry 9 of 23 File: USPT Mar 31, 1998

US-PAT-NO: 5734721
DOCUMENT-IDENTIFIER: US 5734721 A
TITLE: Anti-spoof without error extension (ANSWER)

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 18. Document ID: US 5646676 A Relevance Rank: 68

Entry 12 of 23 File: USPT Jul 8, 1997

US-PAT-NO: 5646676
DOCUMENT-IDENTIFIER: US 5646676 A
TITLE: Scalable interactive multimedia server system for providing on demand data

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 19. Document ID: US 5163131 A Relevance Rank: 44

Entry 22 of 23 File: USPT

US-PAT-NO: 5163131
DOCUMENT-IDENTIFIER: US 5163131 A
TITLE: Parallel I/O network file server architecture

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 20. Document ID: US 5355453 A Relevance Rank: 44

values of any convenient length, including as small as a single bit per use. A random number of arbitrary size may be constructed by concatenating values produced by random number generator **542**. A cryptographically strong pseudorandom sequence may be generated from a random key and seed generated with random number generator **542** and repeated encryption either with the encrypt/decrypt engine **522** or cryptographic algorithms in SPU **500**. Such sequences may be used, for example, in private headers to frustrate efforts to determine an encryption key through cryptoanalysis.

Arithmetic Accelerator **544**

An optional arithmetic accelerator **544** may be provided within an SPU **500** in the form of hardware circuitry that can rapidly perform mathematical calculations such as multiplication and exponentiation involving large numbers. These calculations can, for example, be requested by microprocessor **520** or encrypt/decrypt engine **522**, to assist in the computations required for certain asymmetric encryption/decryption operations. Such arithmetic accelerators are well-known to those skilled in the art. In some implementations, a separate arithmetic accelerator **544** may be omitted and any necessary calculations may be performed by microprocessor **520** under software control.

DMA Controller **526**

DMA controller **526** controls information transfers over address/data bus **536** without requiring microprocessor **520** to process each individual data transfer. Typically, microprocessor **520** may write to DMA controller **526** target and destination addresses and the number of bytes to transfer, and DMA controller **526** may then automatically transfer a block of data between components of SPU **500** (e.g., from ROM **532** to RAM **534**, between encrypt/decrypt engine **522** and RAM **534**, between bus interface unit **530** and RAM **534**, etc.). DMA controller **526** may have multiple channels to handle multiple transfers simultaneously. In some implementations, a separate DMA controller **526** may be omitted, and any necessary data movements may be performed by microprocessor **520** under software control.

Bus Interface Unit (BIU) **530**

Bus interface unit (BIU) **530** communicates information between SPU **500** and the outside world across the security barrier **502**. BIU **530** shown in FIG. **9** plus appropriate driver software may comprise the "appliance link" **510** shown in FIG. **6**. Bus interface unit **530** may be modelled after a USART or PCI bus interface in the preferred embodiment. In this example, BIU **530** connects SPU **500** to electronic appliance system bus **653** shown in FIG. **8**. BIU **530** is designed to prevent unauthorized access to internal components within SPU **500** and their contents. It does this by only allowing signals associated with an SPU **500** to be processed by control programs running on microprocessor **520** and not supporting direct access to the internal elements of an SPU **500**.

Memory Management Unit **540**

Memory Management Unit (MMU) **540**, if present, provides hardware support for memory management and virtual memory management functions. It may also provide heightened security by enforcing hardware compartmentalization of the secure execution space (e.g., to prevent a less trusted task from modifying a more trusted task). More details are provided below in connection with a discussion of the architecture of a Secure Processing Environment ("SPE") **503** supported by SPU **500**.

MMU **540** may also provide hardware-level support functions related to memory management such as, for example, address mapping.

SPU Memory Architecture

In the preferred embodiment, SPU **500** uses three general kinds of memory:

(1) internal ROM **532**;

(2) internal RAM **534**; and

(3) external memory (typically RAM and/or disk supplied by a host electronic appliance).

The internal ROM **532** and RAM **534** within SPU **500** provide a secure operating environment and execution space. Because of cost limitations, chip fabrication size, complexity and other limitations, it may not be possible to provide sufficient memory within SPU **500** to store all information that an SPU needs to process in a secure manner. Due to the practical limits on the amount of ROM **532** and RAM **534** that may be included within SPU **500**, SPU **500** may store information in memory external to it, and move this information into and out of its secure internal memory space on an as needed basis. In these cases, secure processing steps performed by an SPU typically must be segmented into small, securely packaged elements that may be "paged in" and "paged out" of the limited available internal memory space. Memory external to an SPU **500** may not be secure. Since the external memory may not be secure, SPU **500** may encrypt and cryptographically seal code and other information before storing it in external memory. Similarly, SPU **500** must typically decrypt code and other information obtained from external memory in encrypted form before processing (e.g., executing) based on it. In the preferred embodiment, there are two general approaches used to address potential memory limitations in a SPU **500**. In the first case, the small, securely packaged elements represent information contained in secure database **610**. In the second case, such elements may represent protected (e.g., encrypted) virtual memory pages. Although virtual memory pages may correspond to information elements stored in secure database **610**, this is not required in this example of a SPU memory architecture.

The following is a more detailed discussion of each of these three SPU memory resources.

SPU Internal ROM

SPU **500** read only memory (ROM) **532** or comparable purpose device provides secure internal non-volatile storage for certain programs and other information. For example, ROM **532** may store "kernel" programs such as SPU control firmware **508** and, if desired, encryption key information and certain fundamental "load modules." The "kernel" programs, load module information, and encryption key information enable the control of certain basic functions of the SPU **500**. Those components that are at least in part dependent on device configuration (e.g., POST, memory allocation, and a dispatcher) may be loaded in ROM **532** along with additional load modules that have been determined to be required for specific installations or applications.

In the preferred embodiment, ROM **532** may comprise a combination of a masked ROM **532*a*** and an EEPROM and/or equivalent "flash" memory **532*b***. EEPROM or flash memory **532*b*** is used to store items that need to be updated and/or initialized, such as for example, certain encryption keys. An additional benefit of providing EEPROM and/or flash memory **532*b*** is the ability to optimize any load modules and library functions persistently stored within SPU **500** based on typical usage at a specific site. Although these items could also be stored in NVRAM **534*b***, EEPROM and/or flash memory **532*b*** may be more cost effective.

Masked ROM **532*a*** may cost less than flash and/or EEPROM **532*b***, and can be used to store permanent portions

Entry 18 of 23 File: USPT Oct 11, 1994

US-PAT-NO: 5355453
DOCUMENT-IDENTIFIER: US 5355453 A
TITLE: Parallel I/O network file server architecture

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

Generate Collection

Terms	Documents
(copy protect$) and cryptograph$ and (control$ with copy$ with function$)	23

Display 10 Documents including document number 21

Display Format: TI Change Format

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Help | Logout

power supply as the only power source for RTC **528** may significantly reduce the usefulness of time based security techniques unless, at minimum, SPU **500** recognizes any interruption (or any material interruption) of the supply of external power, records such interruption, and responds as may be appropriate such as disabling the ability of the SPU **500** to perform certain or all VDE processes. Recognizing a power interruption may, for example, be accomplished by employing a circuit which is activated by power failure. The power failure sensing circuit may power another circuit that includes associated logic for recording one or more power fail events. Capacitor discharge circuitry may provide the necessary temporary power to operate this logic. In addition or alternatively, SPU **500** may from time to time compare an output of RTC **528** to a clock output of a host electronic appliance **600**, if available. In the event a discrepancy is detected, SPU **500** may respond as appropriate, including recording the discrepancy and/or disabling at least some portion of processes performed by SPU **500** under at least some circumstances

If a power failure and/or RTC **528** discrepancy and/or other event indicates the possibility of tampering, SPU **500** may automatically destroy, or render inaccessible without privileged intervention, one or more portions of sensitive information it stores, such as execution related information and/or encryption key related information. To provide further SPU operation, such destroyed information would have to be replaced by a VDE clearinghouse, administrator and/or distributor, as may be appropriate. This may be achieved by remotely downloading update and/or replacement data and/or code. In the event of a disabling and/or destruction of processes and/or information as described above, the electronic appliance **600** may require a secure VDE communication with an administrator, clearinghouse, and/or distributor as appropriate in order to reinitialize the RTC **528**. Some or all secure SPU **500** processes may not operate until then.

It may be desirable to provide a mechanism for setting and/or synchronizing RTC **528**. In the preferred embodiment, when communication occurs between VDE electronic appliance **600** and another VDE appliance, an output of RTC **528** may be compared to a controlled RTC **528** output time under control of the party authorized to be "senior" and controlling. In the event of a discrepancy, appropriate action may be taken, including resetting the RTC **528** of the "junior" controlled participant in the communication.

SPU Encrypt/Decrypt Engine **522**

In the preferred embodiment, SPU encrypt/decrypt engine **522** provides special purpose hardware (e.g., a hardware state machine) for rapidly and efficiently encrypting and/or decrypting data. In some implementations, the encrypt/decrypt functions may be performed instead by microprocessor **520** under software control, but providing special purpose encrypt/decrypt hardware engine **522** will, in general, provide increased performance. Microprocessor **520** may, if desired, comprise a combination of processor circuitry and dedicated encryption/decryption logic that may be integrated together in the same circuitry layout so as to, for example, optimally share one or more circuit elements.

Generally, it is preferable that a computationally efficient but highly secure "bulk" encryption/decryption technique should be used to protect most of the data and objects handled by SPU **500**. It is preferable that an extremely secure encryption/decryption technique be used as an aspect of authenticating the identity of electronic appliances **600** that are establishing a communication channel and securing any transferred permission, method, and administrative information. In the preferred embodiment, the encrypt/decrypt engine **522** includes both a symmetric key encryption/decryption circuit (e.g. DES, Skipjack/Clipper, IDEA, RC-2, RC-4, etc.) and an antisymmetric (asymmetric) or Public Key ("PK") encryption/decryption circuit The public/private key encryption/decryption circuit is used principally as an aspect of secure communications between an SPU **500** and VDE administrators, or other electronic appliances **600**, that is between VDE secure subsystems. A symmetric encryption/decryption circuit may be used for "bulk" encrypting and decrypting most data stored in secondary storage **662** of electronic appliance **600** in which SPU **500** resides. The symmetric key encryption/decryption circuit may also be used for encrypting and decrypting content stored within VDE objects **300**.

DES or public/private key methods may be used for all encryption functions. In alternate embodiments, encryption and decryption methods other than the DES and public/private key methods could be used for the various encryption related functions. For instance, other types of symmetric encryption/decryption techniques in which the same key is used for encryption and decryption could be used in place of DES encryption and decryption. The preferred embodiment can support a plurality of decryption/encryption techniques using multiple dedicated circuits within encrypt/decrypt engine **522** and/or the processing arrangement within SPU **500**.

Pattern Matching Engine **524**

Optional pattern matching engine **524** may provide special purpose hardware for performing pattern matching functions. One of the functions SPU **500** may perform is to validate/authenticate VDE objects **300** and other items. Validation/authentication often involves comparing long data strings to determine whether they compare in a predetermined way. In addition, certain forms of usage (such as logical and/or physical (contiguous) relatedness of accessed elements) may require searching potentially long strings of data for certain bit patterns or other significant pattern related metrics. Although pattern matching can be performed by SPU microprocessor **520** under software control, providing special purpose hardware pattern matching engine **524** may speed up the pattern matching process.

Compression/Decompression Engine **546**

An optional compression/decompression engine **546** may be provided within an SPU **500** to, for example, compress and/or decompress content stored in, or released from, VDE objects **300**. Compression/decompression engine **546** may implement one or more compression algorithms using hardware circuitry to improve the performance of compression/decompression operations that would otherwise be performed by software operating on microprocessor **520**, or outside SPU **500**. Decompression is important in the release of data such as video and audio that is usually compressed before distribution and whose decompression speed is important. In some cases, information that is useful for usage monitoring purposes (such as record separators or other delimiters) is "hidden" under a compression layer that must be removed before this information can be detected and used inside SPU **500**.

Random Number Generator **542**

Optional random number generator **542** may provide specialized hardware circuitry for generating random values (e.g., from inherently unpredictable physical processes such as quantum noise). Such random values are particularly useful for constructing encryption keys or unique identifiers, and for initializing the generation of pseudo-random sequences. Random number generator **542** may produce

WEST

Help | Logout

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Your wildcard search against 2000 terms has yielded the results below

Search for additional matches among the next 2000 terms

Generate Collection

Search Results - Record(s) 21 through 23 of 23 returned.

□ 21. Document ID: US 5802366 A Relevance Rank: 44

```
Entry 8 of 23                    File: USPT                    Sep 1, 1998

US-PAT-NO: 5802366
DOCUMENT-IDENTIFIER: US 5802366 A
TITLE: Parallel I/O network file server architecture
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 22. Document ID: US 5512977 A Relevance Rank: 44

```
Entry 14 of 23                   File: USPT                    Apr 30, 1996

US-PAT-NO: 5512977
DOCUMENT-IDENTIFIER: US 5512977 A
TITLE: Copying machine with encryption function
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

□ 23. Document ID: US 5931918 A Relevance Rank: 44

```
Entry 3 of 23                    File: USPT                    Aug 3, 1999

US-PAT-NO: 5931918
DOCUMENT-IDENTIFIER: US 5931918 A
TITLE: Parallel I/O network file server architecture
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

Generate Collection

Terms	Documents
(copy protect$) and cryptograph$ and (control$ with copy$ with function$)	23

"oversee" performance of the other required methods in a control process. FIG. **46** shows how the required methods/processes **402**, **404**, **406**, and **408** of FIG. **45** can be organized and controlled by a control method **410**. Control method **410** may call, dispatch events, or otherwise invoke the other methods **402**, **404**, **406**, **408** and otherwise supervise the processing performed in response to an "event."

Control methods operate at the level of control sets **906** within PERCs **808**. They provide structure, logic, and flow of control between disparate acquired methods **1000**. This mechanism permits the content provider to create any desired chain of processing, and also allows the specific chain of processing to be modified (within permitted limits) by downstream redistributors. This control structure concept provides great flexibility.

FIG. **47** shows an example of an "aggregate" method **412** which collects METER method **404**, BUDGET method **406** and BILLING method **408** into an "aggregate" processing flow. Aggregate method **412** may, for example, combine various elements of metering, budgeting and billing into a single method **1000**. Aggregate method **412** may provide increased efficiency as a result of processing METER method **404**, BUDGET method **406** and BILLING method **408** aggregately, but may decrease flexibility because of decreased modularity.

Many different methods can be in effect simultaneously. FIG. **48** shows an example of preferred embodiment event processing using multiple METER methods **404** and multiple BUDGET methods **1408**. Some events may be subject to many different required methods operating independently or cumulatively. For example, in the example shown in FIG. **48**, meter method **404***a* may maintain meter trail and meter information records that are independent from the meter trail and meter information records maintained by METER method **404***b*. Similarly, BUDGET method **408***a* may maintain records independently of those records maintained by BUDGET method **408***b*. Some events may bypass BILLING method **408** while nevertheless being processed by meter method **404***a* and BUDGET method **408***a*. A variety of different variations are possible.

REPRESENTATIVE EXAMPLES OF VDE METHODS

Although methods **1000** can have virtually unlimited variety and some may even be user-defined, certain basic "use" type methods are preferably used in the preferred embodiment to control most of the more fundamental object manipulation and other functions provided by VDE **100**. For example, the following high level methods would typically be provided for object manipulation:

OPEN method

READ method

WRITE method

CLOSE method.

An OPEN method is used to control opening a container so its contents may be accessed. A READ method is used to control the access to contents in a container. A WRITE method is used to control the insertion of contents into a container. A CLOSE method is used to close a container that has been opened.

Subsidiary methods are provided to perform some of the steps required by the OPEN, READ, WRITE and/or CLOSE methods. Such subsidiary methods may include the following:

ACCESS method

PANIC method

ERROR method

DECRYPT method

ENCRYPT method

DESTROY content method

INFORMATION method

OBSCURE method

FINGERPRINT method

EVENT method

CONTENT method

EXTRACT method

EMBED method

METER method

BUDGET method

REGISTER method

BILLING method

AUDIT method

An ACCESS method may be used to physically access content associated with an opened container (the content can be anywhere). A PANIC method may be used to disable at least a portion of the VDE node if a security violation is detected. An ERROR method may be used to handle error conditions. A DECRYPT method is used to decrypt encrypted information. An ENCRYPT method is used to encrypt information. A DESTROY content method is used to destroy the ability to access specific content within a container. An INFORMATION method is used to provide public information about the contents of a container. An OBSCURE method is used to devalue content read from an opened container (e.g., to write the word "SAMPLE" over a displayed image). A FINGERPRINT method is used to mark content to show who has released it from the secure container. An event method is used to convert events into different events for response by other methods.

Open

FIG. **49** is a flowchart of an example of preferred embodiment process control steps for an example of an OPEN method **1500**. Different OPEN methods provide different detailed steps. However, the OPEN method shown in FIG. **49** is a representative example of a relatively full-featured "open" method provided by the preferred embodiment. FIG. **49** shows a macroscopic view of the OPEN method. FIGS. **49***a*–**49***f* are together an example of detailed program controlled steps performed to implement the method shown in FIG. **49**.

The OPEN method process starts with an "open event." This open event may be generated by a user application, an operating system intercept or various other mechanisms for capturing or intercepting control. For example, a user application may issue a request for access to a particular content stored within the VDE container. As another example, another method may issue a command.

In the example shown, the open event is processed by a control method **1502**. Control method **1502** may call other methods to process the event. For example, control method **1502** may call an EVENT method **1504**, a METER method **1506**, a BILLING method **1508**, and a BUDGET method **1510**. Not all OPEN control methods necessarily call of these additional methods, but the OPEN method **1500** shown in FIG. **49** is a representative example.

Control method **1502** passes a description of the open event to EVENT method **1504**. EVENT method **1504** may determine, for example, whether the open event is permitted

Display 10 Documents including document number 23

Display Format: TI Change Format

Main Menu Search Form Posting Counts Show S Numbers Edit S Numbers

Help Logout

trail(s) and audit record(s) for storage into the secured database (blocks **2554**, **2556**). AUDIT method **2520** may then retrieve the audit request(s) from the secure database and determine the response method to run to process the request (blocks **2558**, **2560**). AUDIT method **2520** may at this stage send event(s) contained in the request record(s) to the appropriate response method, and generate response record(s) and requests based on this method (blocks **2562**, **2564**). The processing block **2562** may involve a communication to the outside world.

For example, AUDIT method **2520** at this point could call an external process to perform, for example, an electronic funds transfer against the user's bank account or some other bank account. The AUDIT administrative response can, if desired, call an external process that interfaces VDE to one or more existing computer systems. The external process could be passed the user's account number, PIN, dollar amount, or any other information configured in, or associated with, the VDE audit trail being processed. The external process can communicate with non-VDE hosts and use the information passed to it as part of these communications. For example, the external process could generate automated clearinghouse (ACH) records in a file for submittal to a bank. This mechanism would provide the ability to automatically credit or debit a bank account in any financial institution. The same mechanism could be used to communicate with the existing credit card (e.g. VISA) network by submitting VDE based charges against the charge account.

Once the appropriate Audit response record(s) have been generated, AUDIT method **2520** may write an Audit administrative record(s) into an administrative object for communication back to the VDE user node that generated the Audit request (blocks **2566**, **2568**). The AUDIT method **2520** may then save communications and response processing audit information in appropriate audit trail(s) (blocks **2570**, **2572**) before terminating (at terminate point **2574**).

FIG. **44***c* shows an example of steps that may be performed by the AUDIT method **2520** back at the VDE user node upon receipt of the administrative object generated and sent by FIG. **44***b*, block **2566**. The steps **2580**–**2599** shown in FIG. **44***c* are similar to the steps shown in FIG. **43***d* for the REGISTER method **2400** in the "administrative reply" mode. Briefly, these steps involve receiving and extracting appropriate response records from the administrative object (block **2584**), and then processing the received information appropriately to update secure database records and perform any other necessary actions (blocks **2595**, **2596**).

Examples of Event-Driven Content-Based Methods

VDE methods **1000** are designed to provide a very flexible and highly modular approach to secure processing. A complete VDE process to service a "use event" may typically be constructed as a combination of methods **1000**. As one example, the typical process for reading content or other information from an object **300** may involve the following methods:

an EVENT method

a METER method

a BILLING method

a BUDGET method.

FIG. **45** is an example of a sequential series of methods performed by VDE **100** in response to an event. In this example, when an event occurs, an EVENT method **402** may "qualify" the event to determine whether it is significant or not. Not all events are significant. For example, if the EVENT method **1000** in a control process dictates that usage is to be metered based upon number of pages read, then user request "events" for reading less than a page of information may be ignored. In another example, if a system event represents a request to read a certain number of bytes, and the EVENT method **1000** is part of a control process designed to meter paragraphs, then the EVENT method may evaluate the read request to determine how many paragraphs are represented in the bytes requested. This process may involve mapping to "atomic elements" to be discussed in more detail below.

EVENT method **402** filters out events that are not significant with regard to the specific control method involved. EVENT method **402** may pass on qualified events to a METER process **1404**, which meters or discards the event based on its own particular criteria.

In addition, the preferred embodiment provides an optimization called "precheck." EVENT method/process **402** may perform this "precheck" based on metering, billing and budget information to determine whether processing based on an event will be allowed. Suppose, for example, that the user has already exceeded her budget with respect to accessing certain information content so that no further access is permitted. Although BUDGET method **408** could make this determination, records and processes performed by BUDGET method **404** and/or BILLING method **406** might have to be "undone" to, for example, prevent the user from being charged for an access that was actually denied. It may be more efficient to perform a "precheck" within EVENT method **402** so that fewer transactions have to be "undone."

METER method **404** may store an audit record in a meter "trail" UDE **1200**, for example, and may also record information related to the event in a meter UDE **1200**. For example, METER method **404** may increment or decrement a "meter" value within a meter UDE **1200** each time content is accessed. The two different data structures (meter UDE and meter trail UDE) may be maintained to permit record keeping for reporting purposes to be maintained separately from record keeping for internal operation purposes, for example.

Once the event is metered by METER method **404**, the metered event may be processed by a BILLING method **406**. BILLING method **406** determines how much budget is consumed by the event, and keeps records that are useful for reconciliation of meters and budgets. Thus, for example, BILLING method **406** may read budget information from a budget UDE, record billing information in a billing UDE, and write one or more audit records in a billing trail UDE. While some billing trail information may duplicate meter and/or budget trail information, the billing trail information is useful, for example, to allow a content creator **102** to expect a payment of a certain size, and serve as a reconciliation check to reconcile meter trail information sent to creator **102** with budget trail information sent to, for example, an independent budget provider.

BILLING method **406** may then pass the event on to a BUDGET method **408**. BUDGET method **408** sets limits and records transactional information associated with those limits. For example, BUDGET method **408** may store budget information in a budget UDE, and may store an audit record in a budget trail UDE. BUDGET method **408** may result in a "budget remaining" field in a budget UDE being decremented by an amount specified by BILLING method **406**.

The information content may be released, or other action taken, once the various methods **402**, **404**, **406**, **408** have processed the event.

As mentioned above, PERCs **808** in the preferred embodiment may be provided with "control methods" that in effect

WEST

Help | Logout

Main Menu | Search Form | Result Set | ShowS Numbers | Edit S Numbers | Referring Patents

First Hit | Previous Document | Next Document

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

09/091,735

Document Number 1

Entry 1 of 1 File: USPT Sep 5, 1989

US-PAT-NO: 4864494
DOCUMENT-IDENTIFIER: US 4864494 A

TITLE: Software usage authorization system with key for decrypting/re-encrypting/re-transmitting moving target security codes from protected software
DATE-ISSUED: September 5, 1989

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kobus, Jr.; Paul	Phoenix	AZ	N/A	N/A

ASSIGNEE INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Computerized Data Ssytems for Mfg., Inc.	Phoenix	AZ	N/A	N/A	02

APPL-NO: 6/ 842552
DATE FILED: March 21, 1986

INT-CL: [4] H04L 9/00, H04K 9/00, G06F 12/14
US-CL-ISSUED: ~~364/200, 380/4, 380/25, 364/246.6, 340/825.31~~
US-CL-CURRENT: 713/200; 340/825.31, 705/55 713/201
FIELD-OF-SEARCH: ~~364/2MSFile, 364/9MSFile, 380/3,~~ 380/4, 380/25, 380/29, 340/825.31, 340/825.34

REF-CITED:

U.S. PATENT DOCUMENTS

connected to the File Server. This software coexists with other software running in the Work Station memory. Alert calls can be placed though a modem connected to the Work Station to a pager. No provision is made for alert calls to other than a pager. No provision is made for monitoring power, temperature, intruders, water, fire or smoke. Further, no provision is made for a failure in the Work Station used for monitoring should the Work Station's processor lock up or AC power be lost.

Various devices exist to monitor the environment and issue pre-set voice messages by placing phone calls to predefined numbers, when an alarm condition is detected. Two such products are the EnviroCom I and EnviroCom II manufactured by Best Power Technology, Inc. P.O. Box 280, Needah, Wis. 54646. These products monitor AC electrical power; temperature; loud sound levels, which presumable will be a burglar alarm, fire alarm, or smoke alarm; motion; or the presence of water. No provision by these products is made to permit the monitoring of file servers; recording of user spoken voice alert messages; logging failures detected or the results of alert calls placed; detecting the exact progress of an alert calls placed; so that an alert messages will only be delivered after someone or some device has answered the phone and finished speaking; analyzing the frequency of sound levels detected to discern the exact source of the sound level (e.g. sounds produced by a smoke alarm as opposed to those sounds produced by a fire alarm or some other non-alarm, loud sound level source). Finally, the same set: of alert phone numbers are called, regardless of the type of failure encountered and the number of phone numbers the system can handle is limited.

None of the monitoring devices discussed above make any provision to monitor the status of computer jobs being processed by a Work Station, so that alerts may be issued when the job(s) are completed or a job fails. In addition, no provision is made for more than one AC power input source or the ability to automatically switch between multiple AC input power sources should the primary source of AC input power fail, so that a constant source of AC output power can be provided and controlled to an external device, such as a computer Work Station, and multiple sources of power can be monitored directly by the device. Finally, no provision is made to control the functionality of an external device, such as a Work Station, by temporarily cutting AC power to the Work Station and forcing the Work Station to boot and initiate a predetermined procedure.

SUMMARY OF THE INVENTION

The present invention provides a stand alone Unit containing its own microprocessor designed to monitor the environment and control microprocessor based computers to which the Unit is connected. This Unit is designed for connection to a Work Station. The Unit is controlled by a micro-processor based software operating system residing in the Unit and/or software programs installed in the Work Station to which the Unit is connected. The apparatus is capable of (1) detecting if the Work Station fails; (2) monitoring an unlimited number of File Servers connected over a Local Area Network (LAN); (3) monitoring the availability of public utility power or backup reserve power; (4) switching between main public utility power and backup power sources should main power fail or main power be subsequently restored; (5) detecting the presence of loud audible sounds and discerning the type of device generating said sound by analyzing the sound produced by the device's audible siren (e.g. smoke detector, fire alarm, etc.); (6) permitting a person called to listen over the phone line to any loud sounds present during a loud sound alert call; (7) detecting excessive heat or cold; (8) detecting intruders or other alert conditions detected by external monitoring devices, such as a water sensor; (9) placing alert phone calls to pagers or individuals when alert situations are detected via a direct interface between the microprocessor and the phone line, (10) analyzing sound transmitted over the phone line when alert calls are in progress to determine why a call could not be completed; or, when a call has been successfully completed, the party called has stopped speaking, and the alert message should be delivered; (11) permitting persons alerted to remotely listen to the sound level being produced by an audible alarm siren in a case where a loud sound level has been detected; (11) recording what is spoken by the answering party when an alert message is delivered to a person; (12) monitoring the status of user jobs being processed on any microprocessor based computer to determine when the job has been completed or fails; (13) contacting via telephone and delivering user recorded or prerecorded voice messages alert messages to an unlimited number of designated persons should a alert situation be detected; (14) detecting (via the Work Station) when the Unit has failed so that alert calls can optionally be placed to a pager using a user supplied modem; (15) permitting persons called during an alert situation to confirm that the alert call has been received by pressing a touch tone on their telephone; and (16) permitting persons called to remotely abort any pending alert calls to others from being delivered by pressing a touch tone on their telephone during the time an alert message is being delivered.

The Unit has two input AC power receptacles that are designed to obtain electrical power from an AC public utility power receptacle or an external backup power supply. Should public utility power fail, the microprocessor in the Unit would immediately switch to backup power, if present, so as to prevent any interruption in Work Station processing and permit the Unit to immediately detect that main power failed. If a sustained public utility power failure occurs for more than a specified number of seconds, alert phone calls would be placed to user designated persons indicating primary power has failed and an appropriate announcement would be made locally, using the speaker contained in the Unit. Similarly, if all external sources of power should fail, the Unit would rely on its internal rechargeable battery to deliver user designated alert calls indicating all power has failed.

One AC output power receptacle is provided on the Unit. The Work Station plugs into this receptacle to obtain AC power. When necessary, the Unit will cut power to the Work Station for several seconds forcing the Work Station to re-start (i.e. cold boot). During the boot process, a monitoring session would be initiated automatically, as discussed below. Situations that would require power to be cut in this manner are also discussed below.

Two phones jacks are included in the Unit. The first jack permits the Unit to attach to a public utility phone line. The second jack is used to pass through the phone signal to a standard, external, touch-tone telephone.

Two adapter jacks are presently provided in the Unit to provide low voltage electrical power to and receive alerts from optional intruder detection or environmental monitoring devices. Examples of monitoring devices that can be connected to these jacks include a water alarm, motion alarm, fire alarm, and/or an entrance alarm. The security devices may be temporarily disabled at any time by pressing a security (pulse style) switch, herein referred to as a "Watch Dog" switch, located on the back of the Unit.

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3806882	April 1974	Clarke	340/172.5
3976840	August 1976	Cleveland	179/2.
4120030	October 1978	Johnstone	364/200
4168396	September 1979	Best	178/22
4262329	April 1981	Bright	364/200
4278837	July 1981	Best	178/22.09
4430728	February 1984	Beitel	364/900
4433207	February 1984	Best	380/4
4458315	July 1984	Uchenick	364/200
4465901	August 1984	Best	380/4
4471163	September 1984	Donald	178/22.08
4523271	June 1985	Levien	364/200
4525599	June 1985	Curran et al.	380/29
4558176	December 1985	Arnold et al.	380/4
4588991	May 1986	Atalla	380/4
4590470	May 1986	Koenig	340/825.31
4634808	January 1987	Moerder	380/29
4652900	March 1987	Pailen	364/200
4683968	August 1987	Appelbaum et al.	380/4

ART-UNIT: 232
PRIMARY-EXAMINER: Williams, Jr.; Archie E.
ASSISTANT-EXAMINER: Mohamed; Ayni
ATTY-AGENT-FIRM: Ptak; LaValle D.

ABSTRACT:

A computer based function control system is particularly suited for use as a software security device on the highly popular personal computers or a micro-processor driven function. The system includes an encrypted security message uniquely encoded at predetermined locations within the software or function program. The software or function program includes pre-set errors in it to cause failure of execution of the function or software program unless the errors are nulled during operation of the function or software program. A separate electronic key for retrieving, recognizing, decrypting, encrypting, and producing the null signals is connected to the communications port of the computer from which the key draws its power as well as the security message passed from the computer to the key and back to the computer. There is interchange of moving target and validation information between the computer software and the electronic key. This information is transferred via the security message under the cover of encryption and is monitored by the key and the software to insure that operation of the program can be effected only by authorized users of the function or software program (that is those having the key uniquely associated with that program).
19 Claims, 5 Drawing figures

Main Menu | Search Form | Result Set | Show S Numbers | Edit S Numbers | Referring Patents

First Hit | Previous Document | Next Document

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

Help | Logout

SYSTEM AND METHOD FOR MONITORING COMPUTER ENVIRONMENT AND OPERATION

A portion of this disclosure contains material in which copyright is claimed by the applicant. The applicant has no objection to the copying of this material in the course of making copies of the application file or any patents that may issue on the application, but all other rights whatsoever in the copyrighted material are reserved. statement which was inadvertently omitted from the patent application

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to automated computer network monitoring, environment monitoring, and automatic telephone dialing and message transmission apparatus enabling emergency messages to be transmitted.

Millions of microprocessor based computer networks presently are in use. The term "Network" refers to one or more Work Stations connected by communications lines or cables to one or more central computers, herein referred to as "File Servers". In some File Servers magnetic data storage is often divided into segments, hereinafter referred to as "Volumes". A Network permits users to store data files in a central location to reduce the cost of file storage, eliminate the need for duplicate software systems, and permit all users of the Network to access centrally maintained, up-to-date data files. Because data files are centralized, Network users depend on reliable uninterrupted access to file servers to do their jobs. If one or more File Servers within a Network fails, most companies are literally "out of business" until processing is restored.

Most Networks operate 24 hours per day, but are rarely used more than 12 hours per day. During the period that a Network is not in use, failures can occur. Moreover, the Networks are more vulnerable to major damage due to fire, theft, water, temperature fluctuations, or employee sabotage because no one is typically accessible when the damage starts. In such events occur, the problem would not be detected until someone attempted to use the network at the start of the next business day. When the problem is finally detected, it may take several hours or days to restore normal Network processing. Had the problem been detected when it occurred, damages would be minimized and Network processing could be restored sooner, possibly before the next business day began. Accordingly, there is currently an urgent need for a device designed specifically to monitor Networks during non-business hours. Ideally, this device should be designed to be Fail Safe, activate itself automatically, when deemed appropriate by the user, and use available completer resources where possible. Further, the system should be capable of alerting persons responsible for administering the Network of any file server failures, power failures, intruders, fire, water, smoke or excessive temperature changes, or other similar events that require immediate attention.

The device may also be utilized during normal business hours to monitor computer tasks (i.e. jobs) running on a Work Station and place alert calls when a job within a task has been completed or fails.

2. Background Art

COMPAQ corporation has developed a monitoring system consisting of a 32-bit computer interface board designed to be inserted directly into a File Server. The board must be installed into the File Server and obtains its normal operating power from the File Server's internal low voltage power supply. One board is required for each File Server monitored. The system will not function in existing File Servers that do not have 32-bit EISA slots. Environmental monitoring is limited to voltage and internal temperature sensors. No provision is made for intruder, water, fire or smoke detection. Further, no provision is made to directly monitor AC power flowing into the File Server from an external source(s) so that a main AC power failure will be detected. Instead, the Compaq system monitors only DC voltage flowing to the EISA slot where the interface board is installed. Accordingly, should main AC power fail and the File Server continue to operate off of backup reserve power (e.g. battery power), no provision is made to place a power failure alert until after the reserve power and/or File Server fails.

In the Compaq monitoring system, all alert calls are placed via modem circuitry included on the board or all optional external modem using a serial interface provided on the back plate of the board. No speaker is provided on the board to facilitate call monitoring or localized announcements. Alert messages are delivered via a speech synthesizer or touch tones in the case of an alert call to a pager. No provision is made for recording or replaying user spoken voice messages.

When a call is placed by the Compaq System to a person, no audio analysis is performed to detect when a person has answered the phone call and finished speaking. Instead, after the call is placed, the system assumes someone will answer and hear a pre-recorded message requesting that a tone password be entered. This pre-recorded message is delivered as soon as the call rings, which typically results in the answering party initially receiving only part of the intended message. If the requested touch tone password is not entered and detected by the Compaq system within a specified period of time or the pre-recorded message is fully delivered before someone answers the phone, the call will be aborted and retried later. Accordingly, no alert message will be transmitted, for example, if the person called forgets the required password or the call was answered by an answering machine. Similarly, an alert call placed to a paging service will not be delivered unless a required tone is received from the paging service. No provision is made for paging services that do not use the required tone or attempt made to automatically analyze when the pre-recorded paging service voice prompt has ended, so that the required alert message may be transmitted. Finally, no provision is made in the Compaq system to relate the delivery of alert messages to a specific type of failure. Instead, the same set of alert phone numbers are called, regardless of the type of failure encountered and the number of phone numbers the system can handle is limited to 16 phone numbers.

An alarm device, referred to as SYMON, is being sold by Dataprobe Corporation (170 Coolidge Avenue Englewood, N.J.). The system requires a user supplied external monitoring systems to detect all alarm situations. The system is limited to a maximum of eight alarms. An external modem is required to relay alarm messages in digital (ASCII II) text message form only to user pagers. No provision is made for voice transmitted alert messages. Further, no provision is made for the detection of any alarm situations.

Various software-only products have been developed to monitor File Servers. One product is NetAlarm from Avanti Technology in Austin, Tex. and another product is LAN Server Watch from Brightwork Development Inc. in Tinton Falls, N.J. 07724. These products monitor the status of File Servers using software installed on one of the Work Stations

WEST

Help | Logout

Main Menu | Search Form | Result Set | Show S Numbers | Edit S Numbers | Referring Patents

First Hit | Previous Document | Next Document

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

Document Number 1

Entry 1 of 1 File: USPT Apr 6, 1999

US-PAT-NO: 5892900
DOCUMENT-IDENTIFIER: US 5892900 A

TITLE: Systems and methods for secure transaction management and electronic rights protection
DATE-ISSUED: April 6, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ginter; Karl L.	Beltsville	MD	N/A	N/A
Shear; Victor H.	Bethesda	MD	N/A	N/A
Sibert; W. Olin	Lexington	MA	N/A	N/A
Spahn; Francis J.	El Cerrito	CA	N/A	N/A
Van Wie; David M.	Sunnyvale	CA	N/A	N/A

ASSIGNEE INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
InterTrust Technologies Corp.	Sunnyvale	CA	N/A	N/A	02

APPL-NO: 8/ 706206
DATE FILED: August 30, 1996

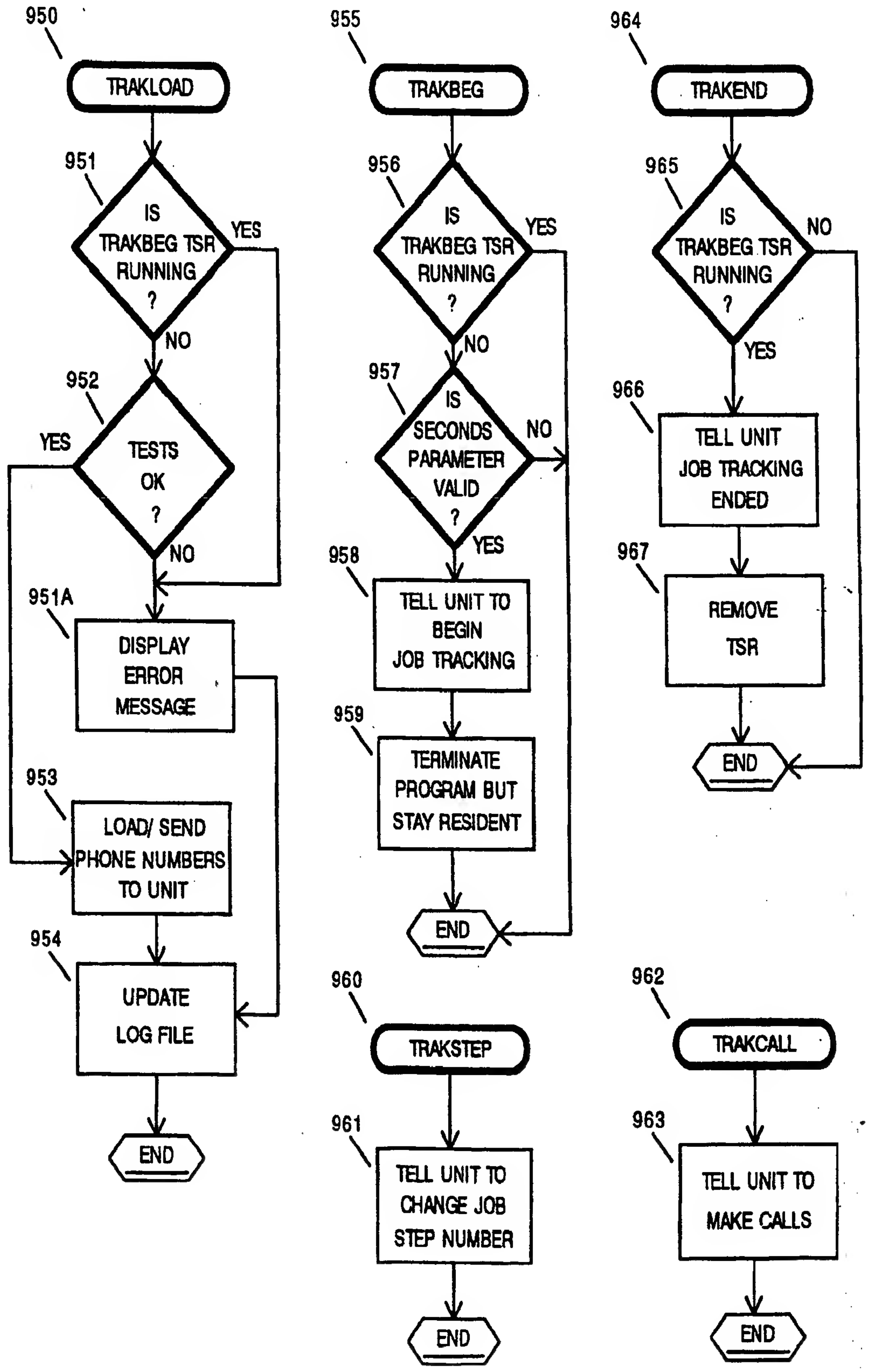
INT-CL: [6] G06 F 11/00
US-CL-ISSUED: 395/186; 395/184.01
US-CL-CURRENT: 713/200; 713/201
FIELD-OF-SEARCH: 395/186, 395/187.01, 395/188.01, 395/218, 395/200.59, 380/4, 380/25, 380/30, 380/825.31, 380/825.34

REF-CITED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
3573747	April 1971	Adams et al.	73/862.58
3609697	September 1971	Blevins	395/407
3796830	March 1974	Smith	380/37
3798359	March 1974	Feistel	380/37
3798360	March 1974	Feistel	380/37
3798605	March 1974	Feistel	380/25
3806882	April 1974	Clarke	711/164
3829833	August 1974	Freeny, Jr.	340/825.31
3906448	September 1975	Henriques	235/438
3911397	October 1975	Freeny, Jr.	235/382
3924065	December 1975	Freeny, Jr.	375/27A
3931504	January 1976	Jacoby	395/186
3946220	March 1976	Brobeck et al.	705/25

FIG. 13



950
TRAKLOAD
951
IS TRAKBEG TSR RUNNING ?
YES
NO
952
TESTS OK ?
YES
NO
951A
DISPLAY ERROR MESSAGE
953
LOAD/ SEND PHONE NUMBERS TO UNIT
954
UPDATE LOG FILE
END
955
TRAKBEG
956
IS TRAKBEG TSR RUNNING ?
YES
NO
957
IS SECONDS PARAMETER VALID ?
NO
YES
958
TELL UNIT TO BEGIN JOB TRACKING
959
TERMINATE PROGRAM BUT STAY RESIDENT
END
960
TRAKSTEP
961
TELL UNIT TO CHANGE JOB STEP NUMBER
END
964
TRAKEND
965
IS TRAKBEG TSR RUNNING ?
NO
YES
966
TELL UNIT JOB TRACKING ENDED
967
REMOVE TSR
END
962
TRAKCALL
963
TELL UNIT TO MAKE CALLS
END

6225059	August 1994	JP
6-215010	August 1994	JP
7-084852	March 1995	JP
7-056794	March 1995	JP
7-141138	June 1995	JP
7-200492	August 1995	JP
7-200317	August 1995	JP
7-244639	September 1995	JP
8-137795	May 1996	JP
8-152990	June 1996	JP
8-185298	July 1996	JP
A2136175	September 1984	GB
2264796	September 1993	GB
2264796A	September 1993	GB
2294348	April 1996	GB
2295947	June 1996	GB
WO A8502310	May 1985	WO
WO 85/03584	August 1985	WO
WO 90/02382	March 1990	WO
WO92/06438	April 1992	WO
WO 92/06438	April 1992	WO
WO92/22870	December 1992	WO
WO 92/22870	December 1992	WO
WO 93/01550	January 1993	WO
WO93/01550	January 1993	WO
WO 94/01821	January 1994	WO
WO94/03859	February 1994	WO
WO 94/03859	February 1994	WO
WO94/06103	March 1994	WO
WO 94/06103	March 1994	WO
WO 94/16395	July 1994	WO
WO 94/18620	August 1994	WO
WO 94/22266	September 1994	WO
WO 94/27406	November 1994	WO
WO 96/00963	January 1996	WO
WO 96/06503	February 1996	WO
WO 96/05698	February 1996	WO
WO 96/03835	February 1996	WO
WO96/13013	May 1996	WO
WO 96/13013	May 1996	WO
WO96/21192	July 1996	WO
WO 96/21192	July 1996	WO
WO 97/03423	January 1997	WO
WO97/07656	March 1997	WO
WO97/32251	September 1997	WO
WO 97/48203	December 1997	WO

ART-UNIT: 275
PRIMARY-EXAMINER: Beausoliel, Jr.; Robert W.
ASSISTANT-EXAMINER: Elisca; Pierre F.
ATTY-AGENT-FIRM: Nixon & Vanderhye P.C.

ABSTRACT:

The present invention provides systems and methods for electronic commerce including secure transaction management and electronic rights

means to place said telephone call when said detecting means detects the presence or absence of said backup AC power.

4. The monitoring apparatus of claim 1 wherein said communication means includes means for transmitting a user-recorded voice message.

5. The monitoring apparatus of claim 3 further comprising:

a mass storage device associated with said computer;

data interface means connected to said control means and said computer for connecting the monitoring apparatus to said computer to transfer data between the monitoring apparatus and said mass storage device associated with said computer in response to signals received from said control means; and

data conversion means connected to said control means and said communication means for receiving digital data defining a user-recorded voice message and generating an output to said communication means compatible with said telephone network, to generate said voice message on said telephone network;

wherein said digital data defining said user recorded voice message is stored on said mass storage device within the computer, and said control means retrieves said digital data and transmits said data to the data conversion means whereby said voice message is generated and transmitted.

6. The monitoring apparatus of claim 1 further comprising recording means connected to the communication means and the control means for recording a response from the person receiving the telephone call.

7. The monitoring apparatus of claim 5 further comprising:

a mass storage device associated with said computer;

data interface means connected to said control means and said computer for connecting the monitoring apparatus to said computer to permit transfer of data between the monitoring apparatus and said mass storage device associated with said computer in response to signals received from said control means;

data conversion means connected to said control means and said communication means for receiving a voice message from said communication means, generating a digital output representative of said voice message, and transmitting said digital output to said control means;

wherein said data interface means transfers said digital output representing said voice message to said computer where said digital data is stored in said mass storage device.

8. A method of monitoring an operation of an external AC power supply system connected to a computer, comprising the steps of:

monitoring power output of said power supply to the computer;

detecting a change in the presence or absence of said power output;

generating an indicating signal when said change in the presence or absence of said power output is detected; and

transmitting said indicating signal to an automatic telephone communications means for selectively establishing a connection between said automatic telephone communications means and a telephone network, and placing a telephone call to a person in a location away from the computer when said change in the presence or absence of said power outant is detected.

9. The method of claim 8 comprising the further steps of:

providing a selectively operable source of backup AC power; and

providing switching means for switching the power supply of the computer from said external AC power supply to said backup AC source in response to said generated indicating signal when said signal indicates that said external AC power is absent.

10. The method of claim 8 comprising the further steps of:

selectively obtaining digital data from a mass storage device associated with said computer, said digital data representing a user-recorded voice message;

converting the digital data to a signal format compatible with said telephone network to form said voice message; and

transmitting said voice message signal format in said telephone call.

11. The method of claim 8 comprising the further steps of:

receiving a voice response signal from the recipient of said telephone call;

converting the voice response signal into a digital data format compatible with a mass storage device associated with said computer and representing said voice response signal; and

storing said digital data in said mass storage device for later retrieval and review.

* * * * *

protection. Electronic appliances such as computers employed in accordance with the present invention help to ensure that information is accessed and used only in authorized ways, and maintain the integrity, availability, and/or confidentiality of the information. Secure subsystems used with such electronic appliances provide a distributed virtual distribution environment (VDE) that may enforce a secure chain of handling and control, for example, to control and/or meter or otherwise monitor use of electronically stored or disseminated information. Such a virtual distribution environment may be used to protect rights of various participants in electronic commerce and other electronic or electronic-facilitated transactions. Secure distributed and other operating system environments and architectures, employing, for example, secure semiconductor processing arrangements that may establish secure, protected environments at each node. These techniques may be used to support an end-to-end electronic information distribution capability that may be used, for example, utilizing the "electronic highway."
220 Claims, 177 Drawing figures

Main Menu | Search Form | Result Set | Show S Numbers | Edit S Numbers | Referring Patents

First Hit | Previous Document | Next Document

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

Help | Logout

UNIT SERIAL CABLE NOT PROPERLY CONNECTED—When this error message occurs, the serial cable between the Unit and the Work Station is properly connected into the serial port.

UNIT NOT RESPONDING—The Unit appears to be properly connected, but the Unit is not responding to requests sent by the Work Station.

If any of the above errors occur, the related error message will appear for approximately 60 seconds; except in the case of the SCHEDULED REBOOT error message, which will appear until a key is pressed. If the F1 Key is pressed during this period, TRAKLOAD will re-attempt processing. This could correct a situation where, for example, the Unit's power ON/OFF switch had just been turned on. When 60 seconds have passed or another key is pressed, batch file processing will continue without the Job Status Monitoring system. Any Job Status Monitoring programs included in the batch file such as TRAKBEG, TRAKCALL, etc. will simply issue the message "TRAKBEG needs to be run first" and then abort.

If no errors are detected 952, processing continues at block 953 where the current Job Status alert dialing string(s) entered during Setup processing 6 are sent to the Unit. Then, a text file named "TRAK.LOG" is updated 954 by the TRAKLOAD program in the default directory used when the TRAKLOAD program is initiated. If this file does not exist, it will be created by the TRAKLOAD program. Otherwise, an entry will be appended to the log indicating when TRAKLOAD was initiated and any errors that may have precluded Job Status Monitoring processing from occurring.

TRAKBEG 955 is the second batch file program executed to initiate a Job Status Monitoring batch file. TRAKBEG installs a small memory resident TSR program (i.e. less than 2K in size) that must be loaded immediately after TRAKLOAD. TRAKBEG processing will terminate if either TRAKLOAD has not been run or TRAKBEG was previously loaded into memory 956. TRAKBEG is automatically removed from memory when the TRAKEND program is invoked at the end of a Job Status Monitoring session. TRAKBEG monitors all disk activity during a Job Status Monitoring session. Optionally, a parameter can be given to TRAKBEG on the program execution command line to set the maximum number of seconds that may occur between successive disk accesses before a job step is considered to have failed 957. The default number of seconds is currently set to 300 seconds. The maximum number of seconds that may be specified is currently set to no more than 999 seconds. If the number specified exceeds this limit or is not a valid numeric expression, TRAKBEG processing is aborted 957. Otherwise, the program notifies the Unit that a job status monitoring session is about to begin 958. When this message is received by the Unit, the Unit expects periodic (i.e. no more than once every five seconds) communications. from the Work Station each time a disk access occurs during the 5 second interval of reporting. If such constant communication cease for longer than the specified maximum number of seconds, the Unit will initiate job step failure calls. As the final step in TRAKBEG processing, TSR is loaded into memory 959 whose function is to monitor disk accesses occurring within the Work Station, as discussed above.

During a monitoring session, the TRAKSTEP program 960 may be inserted into the Job Status Monitoring batch file to assign a numeric ID to the next job(s) being executed. TRAKSTEP must always have a parameter indicating a NUMERIC step number. This step number cannot exceed 2 numeric digits. If the step number exceeds two digits, only the left most two digits will be used as the step number. If a alert or failure call is placed during Job Status Monitoring, step numbering helps the person called determine how far all jobs have progressed. The Job Status Monitoring system does not require that the TRAKSTEP programs be used. The default step assumed by the system when the TRAKSTEP feature is not used is step 1. When TRAKSTEP is executed, the job number specified on the command line is sent to Unit 961 where it is stored in nonvolatile ram and used for any subsequent alert calls.

The next program used for Job Status Monitoring is TRAKCALL 962. Whenever the TRAKCALL program is invoked the TRAKCALL program tells the Unit to place alert calls to everyone contained in the Job status call table indicating the last step number stored in non-volatile RAM was successfully completed 963.

The final program in a Job Status Monitoring session is TRAKEND 964. This program should always be run as the last step in a Job Status Monitoring session. If the Job Status monitoring program is not present in memory 965, TRAKEND has nothing to do and processing is terminated. Otherwise, TRAKEND informs the Unit that the Job Status Monitoring session has ended 966 and no further communications should be expected from the Work Station. Finally, TRAKEND terminates removing the TRAKBEG TSR from memory 967.

We claim:

1. A monitoring apparatus for monitoring an operation of a power supply system connected to a computer, comprising:

first connecting means for connecting an external AC power source to the monitoring apparatus;

power output means connected to said first connecting means for supplying said AC power from the monitoring apparatus to said computer;

detecting means connected to said first connecting means for detecting the presence or absence of said AC power and generating an indicating signal in response;

communication means connected to said detecting means for selectively connecting the monitoring apparatus to a telephone network and placing a telephone call to a person in a location away from the system; and

control means connected to said detecting means and said communication means for receiving said indicating signal and activating said communication means to place said telephone call when said detecting means detects the presence or absence of said AC power.

2. The monitoring apparatus of claim 1 further comprising:

second connecting means for selectively connecting a source of backup AC power to said power output means; and

switching means connected to said first and second connecting means, said detecting means, and said power output means for switching the supply of said power output means from said external AC power source to said backup AC power source in response to said indicating signal from the detecting means when said signal indicates that said external AC power is absent.

3. The monitoring apparatus of claim 2 further comprising second detecting means connected to said second connecting means and said control means for detecting the presence or absence of said backup AC power and generating a second indicating signal in response;

wherein said control means further receives said second indicating signal and activate said communication

WEST

Help Logout

Main Menu | Search Form | Result Set | ShowS Numbers | Edit S Numbers | Referring Patents

First Hit | Previous Document | Next Document

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

Document Number 1

```
Entry 1 of 1                 File: USPT                 Aug 10, 1993

US-PAT-NO: 5235642
DOCUMENT-IDENTIFIER: US 5235642 A

TITLE: Access control subsystem and method for distributed computer
system using locally cached authentication credentials
DATE-ISSUED: August 10, 1993
```

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wobber; Edward	Menlo Park	CA	N/A	N/A
Abadi; Martin	Palo Alto	CA	N/A	N/A
Birrell; Andrew	Los Altos	CA	N/A	N/A
Lampson; Butler	Cambridge	MA	N/A	N/A

ASSIGNEE INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Digital Equipment Corporation	Maynard	MA	N/A	N/A	02

APPL-NO: 7/ 917767
DATE FILED: July 21, 1992

INT-CL: [5] H04K 1/00
US-CL-ISSUED: 380/25; 380/4
US-CL-CURRENT: 713/156; 713/158, 713/164
FIELD-OF-SEARCH: 380/23, 380/25, 380/4

REF-CITED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4677552	June 1987	Sibley	380/23
4847902	July 1989	Hampson	380/25
4941176	July 1990	Matyas et al.	380/25
5032979	July 1991	Hecht et al.	380/23
5081678	January 1992	Kaufman et al.	380/25
5144659	September 1992	Jones	380/23

ART-UNIT: 222
PRIMARY-EXAMINER: Cain; David
ATTY-AGENT-FIRM: Flehr, Hohbach, Test, Albritton & Herbert

ABSTRACT:

A distributed computer system has a number of computers coupled thereto at distinct nodes. The computer at each node of the distributed system has a trusted computing base that includes an authentication agent for authenticating requests received from principals at other nodes in the

sign (#) is generated and sent to the paging service **715**, the telephone is placed back on hook **714**, processing for this routine ends, and ANSWERED is returned to the calling program. If the Fail Safe alert call is being placed as a result of a either a total power failure **708** or a work station failure **710**, the call must be as a result of a loud sound detected. In this case a touch tone is generated representing the number 3, if the loud sound is identified as a son-alert, or 4, if the loud sound could not be specifically identified. Then, the applicable tone is sent to the paging service **713**, a pound sign (#) is generated and sent to the paging service **715** and the telephone is placed back on hook **714**. Then, processing for this routine ends, and ANSWERED is returned to the calling program.

If the call is not being placed as a result of a job tracking monitoring session **707**A, processing continues at block **712B**. If the call is as a result of a job step ending **712B**, the touch tone sequence "*0*" is transmitted to the paging service **712C**. If the call is as a result of a job step failing **712D**, the touch tone sequence "*97*" is transmitted to the paging service **712E**. If the call is as a result of a main power failing **712F**, the touch tone sequence "*96*" is transmitted to the paging service **712G**. If the call is as a result of a Work Station failure **712H**, the touch tone sequence "*98*" is transmitted to the paging service **712I**. If the call is as a result of a Work Station failure **712J**, the touch tone sequence "*99*" is transmitted to the paging service **712K**. Next, DTMF tones corresponding to the job step number active when the call was placed to the paging service are generated and transmitted to the paging service **712L**. If no job step has been specified for the current step in progress, the default code (i.e. "*1*") will be transmitted. If all jobs steps are done by the time the call is placed, the code "*0*" will be transmitted.

After the necessary job status tracking paging codes have been transmitted to the paging service, a pound sign (#) is generated and sent to the paging service **715**, the telephone is placed back on book **714**, processing for this routine ends, and ANSWERED is returned to the calling program.

The process of checking for a dial tone is detailed in the Check For Dial Tone sub-routine starting on FIG. **5AL** beginning at block **716**. This sub-routine resets the call progress timer **716A** and waits up to five seconds **717** & **718** for 1.5 seconds of continuous sound (i.e. the CP-On-Flag remains on for 1.5 seconds), as determined by the Slow-Interrupt routine which constantly updates the CP-On-Flag based on the presence of sound on the phone line **719** to **721**. If the 100's timer reaches 1.5 seconds of continuous sound **720**, a dial tone is presumed to be detected, processing for the routine ends, and "OK" is returned to the calling program. However, if the 20's timer reaches 5 seconds without detecting a condition of at least 1.5 seconds of continuous sound **718**, it is assumed a dial tone was not detected, processing for the routine ends and FAIL is returned to the calling program.

The process of dialing a phone number is detailed in the Dial Phone Number sub-routine on FIG. **5AM** beginning at block **722**. This sub-routine begins by setting a Temporary flag **722A** which is used to indicate when a phone number has not yet been fully dialed. Then, the Unit generates a touch tone dialing digit **723** using digitally stored touch tone wave forms in program memory via the Fast-Interrupt routine. (Note: DTMF telephone touch tone wave forms samples were generated by writing a software Utility incorporating floating point math algorithms). Then, this sub-routine tests for any remaining digits to be dialed **724**. If there are no more digits **725** and the Temporary flag is not

set **730**, the sub-routine will end and return an answered code to the calling program. If the Temporary flag is set **730**, then the Get Call Progress sub-routine is invoked (see FIG. 5AN connector BJ) **731** which returns the status of the call, processing for this routine ends and returns to the calling program. If the flag is not set, processing for the routine ends and ANSWERED is returned to the calling program.

If there are more digits to be processed **725**, and the current digit is not an at-sign '@' **726**, then this subroutine loops back to **723** to dial the next digit. However, if it is an at-sign (meaning the dialing digits necessary to complete the call have been dialed), then the Temporary flag is cleared **727** and the Get-Call-Progress sub-routine **728** is invoked (see FIG. 5AN connector BJ). If the call is answered **729**, this sub-routine loops back to block **724** to process any remaining dialing digits, which represents the desired alert code, an automated switchboard phone extension, etc. that user wants to have delivered after the call is answered. In cases where a call is made to an automated switch board where multiple levels of prerecorded voice messages are announced and touch tones must be entered during (or after) each level's pre-recorded message; multiple '@' symbols and commas (which cause a 2 second pause per comma enter to occur) can be entered as part of the dialing string causing this routine play the required touch tones when necessary and wait until the appropriate point in the voice message system to deliver the alert message. When there are no more digits remaining **725**, the sub-routine will end and return an answered code to the calling program. (Note: at this point the temporary flag will have been set to off in block **727**.)

The Get Call Progress sub-routine (FIG. **5AN**, beginning at block **732**) employs the Unit's stand alone processing capabilities to monitor the status of a Fail Safe calls after the phone number has been dialed.

The rules for call progress determination are as follows. The first sound detected is discarded, which is normally the first phone ring. The routine waits up to ten seconds for this initial sound. A minimum pulse of sound (between 0.1 and 0.2 seconds) is needed so that line clicks and static are ignored. If no initial minimum pulse of sound is detected, the phone call is treated as incomplete. But, if this initial minimum pulse of sound is detected, the system waits during the 10 seconds period for the initial sound to end. If the initial sound does not stop by the end of the 10 second period, the call is treated as uncompleted. This may occur, for example, if the telephone system dropped the call and the telephone system returned to a dial tone.

Once an acceptable initial sound has been discarded, sound "presence" or "silence", is timed and continually tested against set specific limits. There are two counters and a flag associated with this algorithm: a "busy" counter, a "ring" counter and a Voice-Detected flag.

If "sound present" falls between 0.2 and 0.3 seconds (for 0.25 second reorder tone) or 0.45 and 0.55 seconds (for 0.5 second busy tone), then sound present can be narrowed down to either a busy signal or a human voice. Accordingly, the busy counter is incremented and a Voice flag is set. If "sound present" is greater than 0.7 seconds, then the ring counter is incremented. Otherwise, should sound present not satisfy one of these criterion, then voice is considered detected and the busy and ring counters are zeroed out.

If the busy counter reaches a value of ten then the call status is "busy" and the algorithm terminates. (Note: busy signals never have a 0.5 second period of silence).

If the ring counter reaches a value of ten then the call status is "no answer" and the algorithm terminates.

system. Requests are transmitted to servers as messages that include a first identifier provided by the requester and a second identifier provided by the authentication agent of the requester node. Each server process is provided with a local cache of authentication data that identifies requesters whose previous request messages have been authenticated. When a request is received, the server checks the request's first and second identifiers against the entries in its local cache. If there is a match, then the request is known to be authentic. Otherwise, the server node's authentication agent is called to obtain authentication credentials from the requester's node to authenticate the request message. The principal identifier of the requester and the received credentials are stored in a local cache by the server node's authentication agent. The server process also stores a record in its local cache indicating that request messages from the specified requester are known to be authentic, thereby expediting the process of authenticating received requests.

9 Claims, 11 Drawing figures

Main Menu | Search Form | Result Set | Show S Numbers | Edit S Numbers | Referring Patents

First Hit | Previous Document | Next Document

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

Help | Logout

If the command request is to start or end a system configuration session 688, a flag in the Unit indicating a configuration session is activc is turned ON or OFF respec- tively 689. This flag must be set to permit the Unit to discern test alerts occurring during a configuration session from actual alerts, since different alert processing procedures may occur during testing.

If the command request is to make Job Status Monitoring alert calls 690, the Work Station has detected that a job step being monitored has failed and alert calls need to be placed by the Unit. The Unit places all required alert calls using the dialing strings and active job step number stored in the Unit's static RAM then delivers the pre-set job status alert message when a call has been completed 691.

If the command request is to process a specific dial tone 692, the command is followed by the digitized dial tone to converted to sound by the Unit and transferred out over the phone and/or the Units internal speaker 693. This approach permits the Work Station's Interface program to control the dialing of alert phone numbers.

If the command request is to Wait For a touch tone 692A, the Check-DTMF flag is set 692B which will cause the Fast-Interrupt routine to begin to analyze if a specify tone is present on the phone line, indicating an alert call has been confirmed by the person answering the alert call (during the period of silence occurring between each repetition of an alert message delivered).

If the command request is to pass through any telephone sound present on the phone line when an alert call is placed 692C, the Pass-Thru flag will be set on 692D which will cause the Fast-Interrupt routine to begin to pass through the sound occurring after an alert call is placed to the Work Station.

When any of the above commands are received by the Unit, processing ends normally when the command request has been satisfied and a result code indicating what hap- pened when the command was executed is returned to the Interface program in the Work Station. If the command request could not be identified, the Unit returns a NAK indicating the command was not recognized 694.

The process of calling a phone number is detailed in the Call Phone sub-routine starting on FIG. 5AJ beginning at block 695. Processing begins by taking the phone line off hook (i.e. switch open) 695A so that a call can be placed. The Check-For-Dial Tone sub-routine (FIG. 5AL at connec- tor BG is then called to confirm the presence of a dial tone 696.

If there is no dial tone 697 (i.e. a fail is returned after test 718), then the phone is reset to an on hook status (i.e. switch closed) 698 and an ERROR status, indicating a phone line failure, is returned to the calling routine. Otherwise, the phone number is dialed 699 using the Dial-Number and Get Call Progress subroutines (FIG. 5AM, connector BH. If the call is not answered 700, the phone line is reset to an on hook status 698 and an ERROR status, indicating there was no answer, is returned to the calling program and processing for the sub-routine ends.

If a call is placed to a person 701, several special characters may be specified by the user within the dialing string to activate several optional features. If the dialing string contains an "&" symbol, the number of times that the call is repeated can be changed from the current default of three repetitions to a higher number of repetitions is desired by the user. The specific number of repetitions desired follows immediately after the "&" in the dialing string and ends with an "&" character (e.g. &10& means 10 repetitions). If the dialing string contains a "!" symbol; then, should the user press a touch tone after the call is answered, the call will be considered answered and any other pending alert calls will be discarded. In this casc, the person answer- ing the alert is in effect telling the system not to be concerned about any pending alerts messages. If the dialing string contains a "!" symbol, the call is considered answered, but a touch tone is not detected; then the call will be treated as delivered, but pending alert messages will not be discarded. If the dialing string contains the symbol "%", then the call will not be considered delivered unless a touch tone response from the person called is detected.

If the call is to a person requiring a spoken message 701, a counter is set (i.e. register 4) to contain the number of times the alert message will be repeated 702. If the alert dialing string does not specify the desired number of repetitions using the "&" characters as discussed above, then the present default of three repetitions will be assumed. Then, the Abort-To-Mainloop flag is checked at 703 to determine if any event has occurred, such as someone turning off the Unit that would cause calls to be aborted and this sub-routine to be terminated. If this flag is set, the phone is reset to an on hook status 698, processing for the sub-routine ends and a call aborted ERROR code returned to the calling program. Next, 704 causes the digitally recorded alert message to be spoken in a humanly intelligible form. Then, if the alert call was placed as a result of a loud sound detected, the Pass- Thru flag is enabled 704A to instruct the Fast-Interrupt routine to pass through whatever sounds are detected by the Unit's microphone to the person answering the call. Pres- ently, 5 seconds of sound is passed through in this manner.

If the dialing string called contains an "!" or a "%" character, the DTMF-Detection flag will enabled. If this flag is enabled 704B, the Unit listens for two second during the several seconds of silence between message repetitions for a designated touch tone to be pressed by the person answer- ing the call 704C. If during this sampling period, a specified touch tone is detected (entered by the person answering the alert call) 704D, several audible beeps are send to the phone lille by the Unit to acknowledge the tone has been detected, processing for this sub-routine ends, and RESPONSE is returned to the calling program. Otherwise, if the required touch tone was not detected 704D or the DTMF-Detection flag was not enabled 704B, the Unit waits for several seconds 705 and then decrements the message repetition counter 705. If the message repetition counter is zero, the phone is placed on hook 714, processing for the sub-routine ends, and ANSWERED is returned to the calling program. If the message repetition counter has not yet reached zero 706, processing loops back to block 703 to repeat the message delivery process again.

If a call is placed to an automated paging service 701 several additional code dialing digits are entered into the phone line after the call is complete and any user generated codes are transmitted to the paging service. First, an asterisk touch tone is generated to identify the start of the Unit's alert error type code 707. If the alert call is to a Fail safe dialing string stored in non-volatile RAM processing continues at block 708. If the alert call is being placed as a result of a total power failure 708, a touch tone is generated representing the number 1 is then sent to the paging service 709, a pound sign (#) is generated and sent to the paging service 715, the telephone is placed back on hook 714, processing for this routine ends, and ANSWERED is returned to the calling program. If the alert call is being placed as a result of a Work Station failure 710, a touch tone is generated representing the number 2 is then sent to the paging service 711, a pound